Metrocar Funnel Analysis SQL Queries for Tableau

QUESTION:1 What steps of the funnel should we research and improve? Are there any specific drop-off points preventing users from completing their first ride?

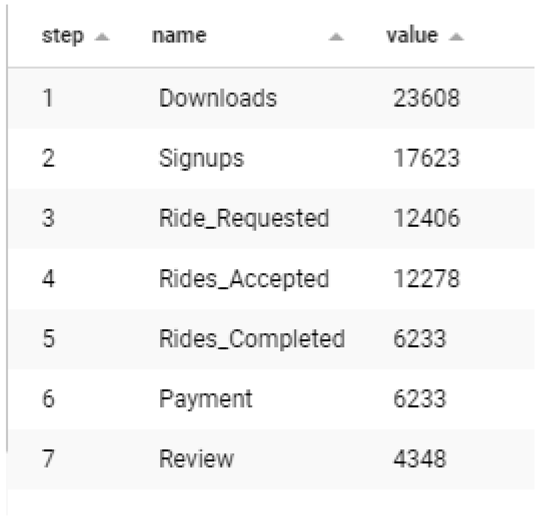
AND

QUESTION 5: What part of our funnel has the lowest conversion rate? What can we do to improve this part of the funnel?

User level:

```
-- Calculate aggregated data for each step
WITH new_table AS (
  SELECT 1 AS step,
        'Downloads' AS name,
        COUNT(DISTINCT app_download_key) AS value
  FROM app_downloads
  UNION
  SELECT 2 AS step,
        'Signups' AS name,
        COUNT(DISTINCT user_id) AS value
  FROM signups
  UNION
  SELECT 3 AS step,
        'Ride_Requested' AS name,
        COUNT(DISTINCT user_id) AS value
  FROM ride_requests
  UNION
  SELECT 4 AS step,
        'Rides_Accepted' AS name,
        COUNT(DISTINCT user_id) AS value
  FROM ride_requests
  WHERE accept_ts IS NOT NULL
  UNION
  SELECT 5 AS step,
        'Rides_Completed' AS name,
        COUNT(DISTINCT user_id) AS value
  FROM ride_requests
  WHERE pickup_ts IS NOT NULL AND dropoff_ts IS NOT NULL
  UNION
  SELECT 6 AS step,
        'Payment' AS name,
        COUNT(DISTINCT r.user_id) AS value
  FROM ride_requests AS r
  INNER JOIN transactions AS t ON r.ride_id = t.ride_id
  WHERE charge_status = 'Approved'
  UNION
  SELECT 7 AS step,
        'Review' AS name,
        COUNT(DISTINCT user_id) AS value
  FROM reviews
  ORDER BY step
)

-- Select all data from the new_table
SELECT
  *
FROM
  new_table;
```

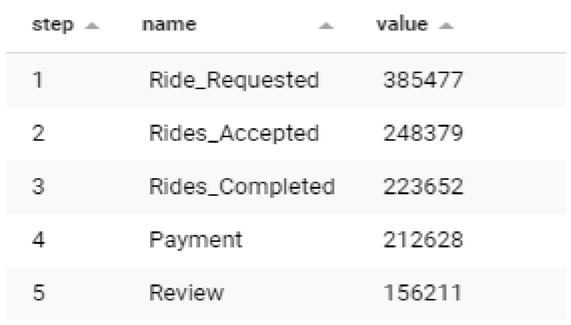


step	name	value
1	Downloads	23608
2	Signups	17623
3	Ride_Requested	12406
4	Rides_Accepted	12278
5	Rides_Completed	6233
6	Payment	6233
7	Review	4348

Ride level:

```
-- Calculate aggregated data for each step
WITH new_table AS (
  SELECT 1 AS step,
        'Ride_Requested' AS name,
        COUNT(DISTINCT ride_id) AS value
  FROM ride_requests
  UNION
  SELECT 2 AS step,
        'Rides_Accepted' AS name,
        COUNT(DISTINCT ride_id) AS value
  FROM ride_requests
  WHERE accept_ts IS NOT NULL
  UNION
  SELECT 3 AS step,
        'Rides_Completed' AS name,
        COUNT(DISTINCT ride_id) AS value
  FROM ride_requests
  WHERE pickup_ts IS NOT NULL AND dropoff_ts IS NOT NULL
  UNION
  SELECT 4 AS step,
        'Payment' AS name,
        COUNT(DISTINCT r.ride_id) AS value
  FROM ride_requests AS r
  INNER JOIN transactions AS t ON r.ride_id = t.ride_id
  WHERE charge_status = 'Approved'
  UNION
  SELECT 5 AS step,
        'Review' AS name,
        COUNT(DISTINCT ride_id) AS value
  FROM reviews
  ORDER BY step
)

-- Calculate the drop-off rate
SELECT
  *
FROM
  new_table;
```



step	name	value
1	Ride_Requested	385477
2	Rides_Accepted	248379
3	Rides_Completed	223652
4	Payment	212628
5	Review	156211

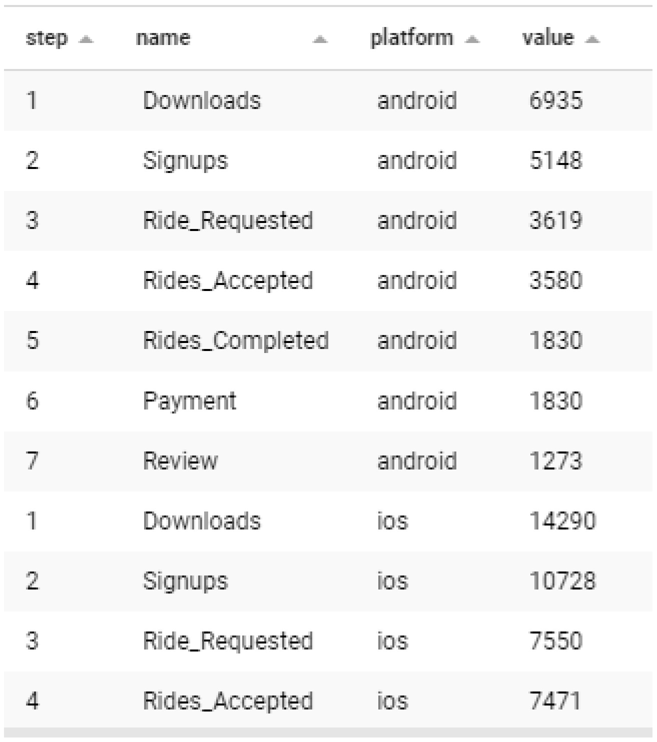
QUESTION 2: **Metrocar currently supports 3 different platforms: ios, android, and web. To recommend where to focus our marketing budget for the upcoming year, what insights can we make based on the platform?**

User Level:

```
-- Common Table Expression (CTE) to compute the aggregated data
WITH new_table AS (
   SELECT a.platform AS platform,
        COUNT(DISTINCT a.app_download_key) AS download_users,
        COUNT(DISTINCT s.user_id) AS signups_users,
        COUNT(DISTINCT r.user_id) AS ride_requested_users,
        COUNT(DISTINCT CASE WHEN r.accept_ts IS NOT NULL THEN r.user_id END) AS
ride_accepted_users,
        COUNT(DISTINCT CASE WHEN r.pickup_ts IS NOT NULL AND r.dropoff_ts IS NOT
NULL THEN r.user_id END) AS ride_completed_users,
        COUNT(DISTINCT CASE WHEN tr.charge_status = 'Approved' THEN r.user_id END) AS
payment_users,
        COUNT(DISTINCT rw.user_id) AS review_users
   FROM app_downloads AS a
   LEFT JOIN signups AS s ON a.app_download_key = s.session_id
   LEFT JOIN ride_requests AS r ON s.user_id = r.user_id
   LEFT JOIN transactions AS tr ON r.ride_id = tr.ride_id
   LEFT JOIN reviews AS rw ON r.user_id = rw.user_id
   GROUP BY a.platform
),

-- Common Table Expression (CTE) to union the data and define steps
union_table AS (
    SELECT 1 AS step,
        'Downloads' AS name,
        platform,
        download_users AS value
    FROM new_table
    UNION
    SELECT 2 AS step,
        'Signups' AS name,
        platform,
        signups_users AS value
    FROM new_table
    UNION
    SELECT 3 AS step,
        'Ride_Requested' AS name,
        platform,
        ride_requested_users AS value
    FROM new_table
    UNION
    SELECT 4 AS step,
        'Rides_Accepted' AS name,
        platform,
        ride_accepted_users AS value
    FROM new_table
    UNION
    SELECT 5 AS step,
        'Rides_Completed' AS name,
        platform,
        ride_completed_users AS value
    FROM new_table
    UNION
    SELECT 6 AS step,
        'Payment' AS name,
        platform,
        payment_users AS value
    FROM new_table
    UNION
    SELECT 7 AS step,
        'Review' AS name,
        platform,
        review_users AS value
    FROM new_table
    ORDER BY platform, step
)
```

```
-- Final query with necessary calculations
SELECT *
FROM union_table;
```



step	name	platform	value
1	Downloads	android	6935
2	Signups	android	5148
3	Ride_Requested	android	3619
4	Rides_Accepted	android	3580
5	Rides_Completed	android	1830
6	Payment	android	1830
7	Review	android	1273
1	Downloads	ios	14290
2	Signups	ios	10728
3	Ride_Requested	ios	7550
4	Rides_Accepted	ios	7471

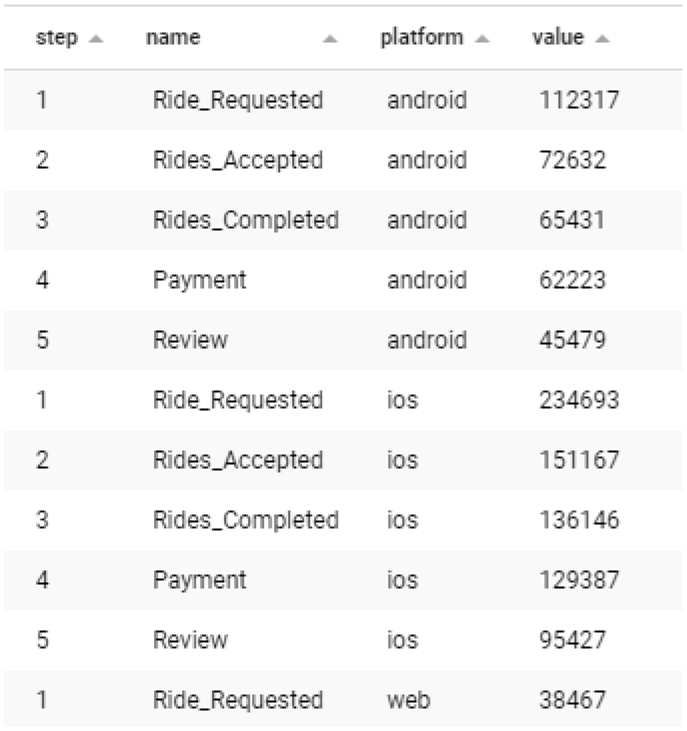
Ride Level:

```
-- Common Table Expression (CTE) to compute the aggregated data
WITH new_table AS (
   SELECT a.platform AS platform,
          COUNT(DISTINCT r.ride_id) AS total_ride_requested,
          COUNT(DISTINCT CASE WHEN r.accept_ts IS NOT NULL THEN r.ride_id END) AS
total_ride_accepted,
          COUNT(DISTINCT CASE WHEN r.pickup_ts IS NOT NULL AND r.dropoff_ts IS NOT
NULL THEN r.ride_id END) AS total_ride_completed,
          COUNT(DISTINCT CASE WHEN tr.charge_status = 'Approved' THEN r.ride_id END) AS
ride_payment,
          COUNT(DISTINCT rw.ride_id) AS ride_review
   FROM app_downloads AS a
   LEFT JOIN signups AS s ON a.app_download_key = s.session_id
   LEFT JOIN ride_requests AS r ON s.user_id = r.user_id
```

```
    LEFT JOIN transactions AS tr ON r.ride_id = tr.ride_id
    LEFT JOIN reviews AS rw ON r.user_id = rw.user_id
    GROUP BY a.platform
),

-- Common Table Expression (CTE) to union the data and define steps
union_table AS (
    SELECT 1 AS step,
           'Ride_Requested' AS name,
           platform,
           total_ride_requested AS value
    FROM new_table
    UNION
    SELECT 2 AS step,
           'Rides_Accepted' AS name,
           platform,
           total_ride_accepted AS value
    FROM new_table
    UNION
    SELECT 3 AS step,
           'Rides_Completed' AS name,
           platform,
           total_ride_completed AS value
    FROM new_table
    UNION
    SELECT 4 AS step,
           'Payment' AS name,
           platform,
           ride_payment AS value
    FROM new_table
    UNION
    SELECT 5 AS step,
           'Review' AS name,
           platform,
           ride_review AS value
    FROM new_table
    ORDER BY platform, step
)

-- Final query with necessary calculations
SELECT *
FROM union_table;
```



step	name	platform	value
1	Ride_Requested	android	112317
2	Rides_Accepted	android	72632
3	Rides_Completed	android	65431
4	Payment	android	62223
5	Review	android	45479
1	Ride_Requested	ios	234693
2	Rides_Accepted	ios	151167
3	Rides_Completed	ios	136146
4	Payment	ios	129387
5	Review	ios	95427
1	Ride_Requested	web	38467

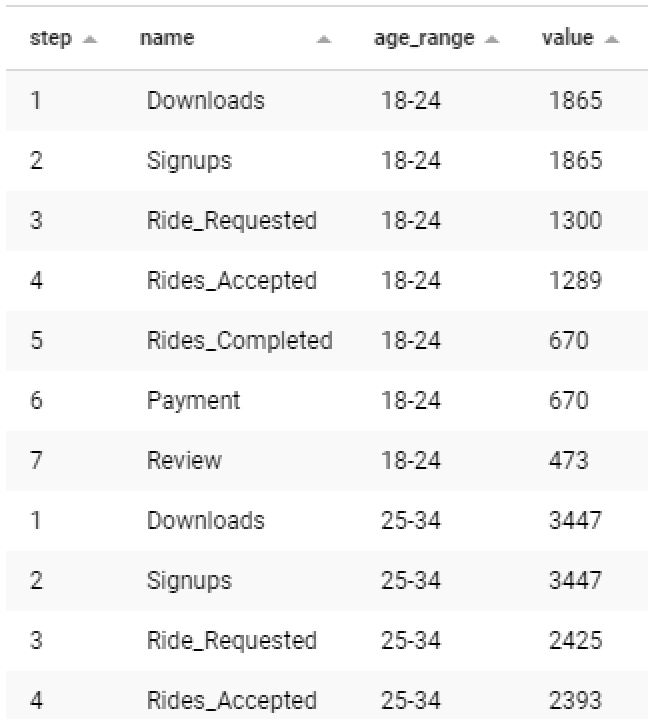
QUESTION 3: What age groups perform best at each stage of our funnel? Which age group(s) likely contain our target customers?

User Level:

```
-- Define CTE new_table to compute counts for each age range and each stage
WITH new_table AS (
   SELECT s.age_range AS age_range,
         COUNT(DISTINCT a.app_download_key) AS download_users,
         COUNT(DISTINCT s.user_id) AS signups_users,
         COUNT(DISTINCT r.user_id) AS ride_requested_users,
         COUNT(DISTINCT CASE WHEN r.accept_ts IS NOT NULL THEN r.user_id END) AS
ride_accepted_users,
         COUNT(DISTINCT CASE WHEN r.pickup_ts IS NOT NULL AND r.dropoff_ts IS NOT
NULL THEN r.user_id END) AS ride_completed_users,
         COUNT(DISTINCT CASE WHEN tr.charge_status = 'Approved' THEN r.user_id END) AS
payment_users,
         COUNT(DISTINCT rw.user_id) AS review_users
   FROM app_downloads AS a
   LEFT JOIN signups AS s ON a.app_download_key = s.session_id
   LEFT JOIN ride_requests AS r ON s.user_id = r.user_id
   LEFT JOIN transactions AS tr ON r.ride_id = tr.ride_id
   LEFT JOIN reviews AS rw ON r.user_id = rw.user_id
   GROUP BY s.age_range
),

-- Define CTE union_table to merge results from new_table for each stage
union_table AS (
   SELECT 1 AS step,
         'Downloads' AS name,
         age_range,
         download_users AS value
   FROM new_table
   UNION
   SELECT 2 AS step,
         'Signups' AS name,
         age_range,
         signups_users AS value
   FROM new_table
   UNION
   SELECT 3 AS step,
        'Ride_Requested' AS name,
        age_range,
        ride_requested_users AS value
    FROM new_table
    UNION
    SELECT 4 AS step,
        'Rides_Accepted' AS name,
        age_range,
        ride_accepted_users AS value
    FROM new_table
    UNION
    SELECT 5 AS step,
        'Rides_Completed' AS name,
        age_range,
        ride_completed_users AS value
    FROM new_table
    UNION
    SELECT 6 AS step,
        'Payment' AS name,
        age_range,
        payment_users AS value
    FROM new_table
    UNION
    SELECT 7 AS step,
        'Review' AS name,
        age_range,
        review_users AS value
    FROM new_table
    ORDER BY age_range, step
)

-- Final query to calculate conversion rate and drop-off rate
SELECT *
FROM union_table;
```

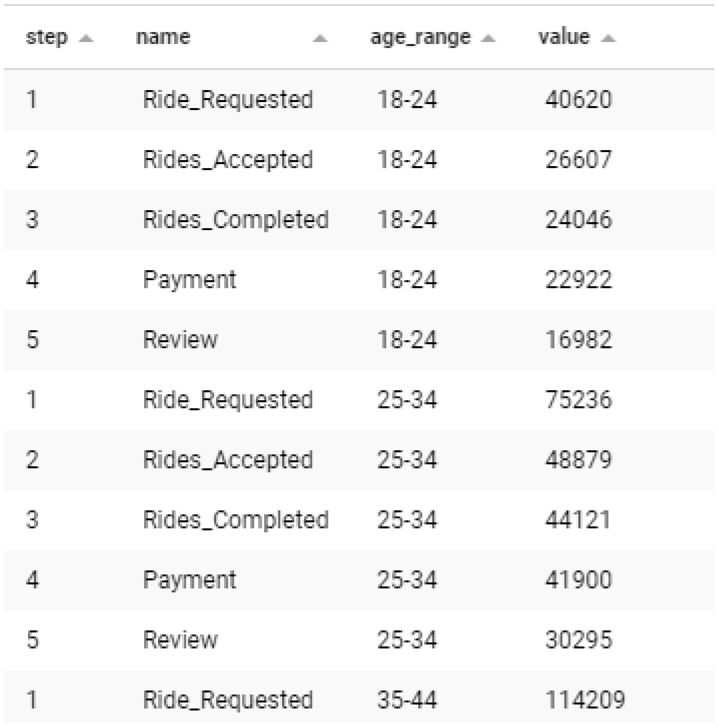


step	name	age_range	value
1	Downloads	18-24	1865
2	Signups	18-24	1865
3	Ride_Requested	18-24	1300
4	Rides_Accepted	18-24	1289
5	Rides_Completed	18-24	670
6	Payment	18-24	670
7	Review	18-24	473
1	Downloads	25-34	3447
2	Signups	25-34	3447
3	Ride_Requested	25-34	2425
4	Rides_Accepted	25-34	2393

Ride Level:

```
-- Define CTE new_table to compute counts for each age range and each stage
WITH new_table AS (
   SELECT s.age_range AS age_range,
         COUNT(DISTINCT r.ride_id) AS total_ride_requested,
         COUNT(DISTINCT CASE WHEN r.accept_ts IS NOT NULL THEN r.ride_id END) AS
total_ride_accepted,
         COUNT(DISTINCT CASE WHEN r.pickup_ts IS NOT NULL AND r.dropoff_ts IS NOT
NULL THEN r.ride_id END) AS total_ride_completed,
         COUNT(DISTINCT CASE WHEN tr.charge_status = 'Approved' THEN r.ride_id END) AS
ride_payment,
         COUNT(DISTINCT rw.ride_id) AS ride_review
   FROM app_downloads AS a
   LEFT JOIN signups AS s ON a.app_download_key = s.session_id
   LEFT JOIN ride_requests AS r ON s.user_id = r.user_id
   LEFT JOIN transactions AS tr ON r.ride_id = tr.ride_id
   LEFT JOIN reviews AS rw ON r.user_id = rw.user_id
   GROUP BY s.age_range
),
-- Define CTE union_table to merge results from new_table for each stage
union_table AS (
   SELECT 1 AS step,
        'Ride_Requested' AS name,
        age_range,
        total_ride_requested AS value
   FROM new_table
   UNION
   SELECT 2 AS step,
        'Rides_Accepted' AS name,
        age_range,
        total_ride_accepted AS value
   FROM new_table
   UNION
   SELECT 3 AS step,
        'Rides_Completed' AS name,
        age_range,
        total_ride_completed AS value
   FROM new_table
   UNION
   SELECT 4 AS step,
        'Payment' AS name,
        age_range,
        ride_payment AS value
   FROM new_table
   UNION
   SELECT 5 AS step,
        'Review' AS name,
        age_range,
        ride_review AS value
   FROM new_table
   ORDER BY age_range, step
)

-- Final query to calculate conversion rate and drop-off rate
SELECT *
FROM union_table;
```

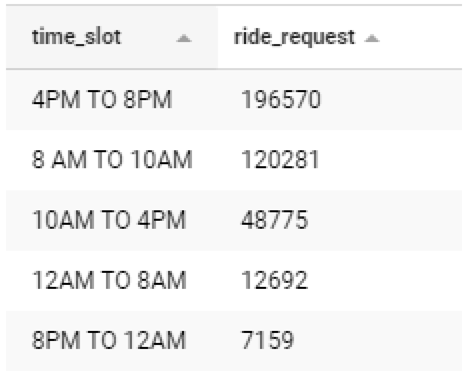


step	name	age_range	value
1	Ride_Requested	18-24	40620
2	Rides_Accepted	18-24	26607
3	Rides_Completed	18-24	24046
4	Payment	18-24	22922
5	Review	18-24	16982
1	Ride_Requested	25-34	75236
2	Rides_Accepted	25-34	48879
3	Rides_Completed	25-34	44121
4	Payment	25-34	41900
5	Review	25-34	30295
1	Ride_Requested	35-44	114209

QUESTION 4: **Surge pricing is the practice of increasing the price of goods or services when there is the greatest demand for them. If we want to adopt a price-surging strategy, what does the distribution of ride requests look like throughout the day?**

Peak Hour Distribution:

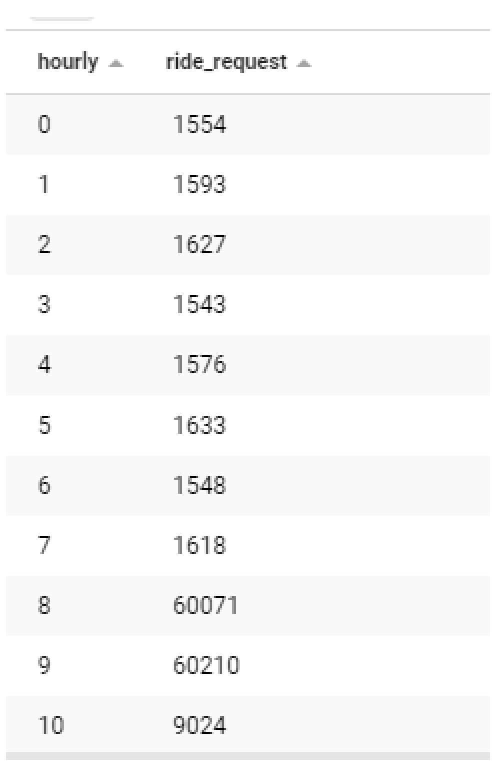
```
SELECT
    -- Categorize time slots based on the hour extracted from request_ts
    CASE
        WHEN EXTRACT(HOUR FROM request_ts) >= 8 AND EXTRACT(HOUR FROM
request_ts) < 10 THEN '8 AM TO 10AM'
        WHEN EXTRACT(HOUR FROM request_ts) >= 10 AND EXTRACT(HOUR FROM
request_ts) < 16 THEN '10AM TO 4PM'
        WHEN EXTRACT(HOUR FROM request_ts) >= 16 AND EXTRACT(HOUR FROM
request_ts) < 20 THEN '4PM TO 8PM'
        WHEN EXTRACT(HOUR FROM request_ts) >= 20 AND EXTRACT(HOUR FROM
request_ts) <= 24 THEN '8PM TO 12AM'
        WHEN EXTRACT(HOUR FROM request_ts) >= 0 AND EXTRACT(HOUR FROM
request_ts) < 8 THEN '12AM TO 8AM'
        ELSE 'time'
    END AS time_slot,
    COUNT(ride_id) AS ride_request
FROM ride_requests
-- Group the results by the time slots
GROUP BY time_slot
-- Order the results by the count of ride requests in descending order
ORDER BY ride_request DESC;
```



time_slot	ride_request
4PM TO 8PM	196570
8 AM TO 10AM	120281
10AM TO 4PM	48775
12AM TO 8AM	12692
8PM TO 12AM	7159

Hourly Distribution:

```
-- Extract the hour from the request timestamp
SELECT EXTRACT(HOUR FROM request_ts) AS hourly,
       -- Count the number of ride requests
       COUNT(ride_id) AS ride_request
FROM ride_requests
GROUP BY EXTRACT(HOUR FROM request_ts)
ORDER BY hourly;
```



hourly	ride_request
0	1554
1	1593
2	1627
3	1543
4	1576
5	1633
6	1548
7	1618
8	60071
9	60210
10	9024